BAB II

LANDASAN TEORI

2.1 Pengertian Hotel

Hotel berasal dari kata hostel, konon diambil dari bahasa Perancis kuno. Bangunan publik ini sudah disebut-sebut sejak akhir abad ke-17. Maknanya kira-kira, "tempat penampungan buat pendatang" atau bisa juga "bangunan penyedia pondokan dan makanan untuk umum". Jadi, pada mulanya hotel memang diciptakan untuk meladeni masyarakat.

Tak aneh kalau di Inggris dan Amerika, yang namanya pegawai hotel dulunya mirip pegawai negeri alias abdi masyarakat. Tapi, seiring perkembangan zaman dan bertambahnya pemakai jasa, layanan inap-makan ini mulai meninggalkan misi sosialnya. Tamu pun dipungut bayaran. Sementara bangunan dan kamar-kamarnya mulai ditata sedemikian rupa agar membuat tamu betah. Meskipun demikian, bertahun-tahun standar layanan hotel tak banyak berubah.

Sampai pada tahun 1793, saat City Hotel dibangun di cikal bakal wilayah kota New York. City Hotel itulah pelopor pembangunan penginapan gaya baru yang lebih fashionable. Sebab, dasar pembangunannya tak hanya mementingkan lokasi yang strategis. Tapi juga pemikiran bahwa hotel juga tempat istirahat yang mumpuni. Jadi, tak ada salahnya didirikan di pinggir kota.

Setelah itu, muncul hotel-hotel legendaris seperti Tremont House (Boston, 1829) yang selama puluhan tahun dianggap sebagai salah satu tempat paling top di Amerika Serikat (AS). Tremont bersaing ketat dengan Astor House, yang dibangun di New York, 1836. Saat itu, hotel modern identik dengan perkembangan lalu lintas dan tempat beristirahat. Saat pembangunan jaringan kereta api sedang gencar-gencarnya, hampir di tiap perhentian (stasiun) ada hotel.

Maksudnya jelas, untuk mengakomodasi orang-orang yang baru saja bepergian dengan kereta api. Karena masa itu naik kereta api sangat melelahkan, hotel-hotel pun "dipersenjatai" berbagai hiburan pelepas penat. Hotel jenis ini, diembeli-embeli dengan kata "transit", karena memang ditujukan buat para musafir.

Seiring dengan berkembangnya teknologi dan makin luasnya jangkauan angkutan darat (terlebih setelah ditemukannya kendaraan bermotor), kawasan sekitar rel kereta api tak lagi menarik minat para investor. Orang kemudian lebih suka jalan-jalan pakai mobil ketimbang kereta. Kepopuleran hotel transit pun tersaingi oleh kehadiran "motel", gabungan kata "motor hotel" yang sama dengan tempat istirahat para pengendara kendaraan bermotor.

Kejayaan motel tak berlangsung lama. Seiring makin pesatnya perkembangan kota, berakhir pula era motel. Terutama karena lokasinya yang agak di pinggir kota dan fasilitasnya yang kalah bagus dengan hotel di pusat

kota. Kalaupun terpaksa bermalam di kawasan pinggiran, motel harus bersaing dengan hotel resort, yang banyak tumbuh di tempat-tempat peristirahatan.

Selain hotel, resort, anak-anak kandung hotel yang lahir di era 1990-an tak kalah hebatnya. Sebut saja berbagai extended-stay hotel, khusus buat tamu yang membutuhkan tempat menginap minimal lima malam. Sedangkan pelaku bisnis yang harus bernegosiasi di kampung atau negeri orang, bisa mencari hotel apartment. Di Amerika, dua jenis hotel ini berkembang sangat pesat.

Di Indonesia, kata hotel selalu dikonotasikan sebagai bangunan penginapan yang cukup mahal. Umumnya di Indonesia dikenal hotel berbintang, hotel melati yang tarifnya cukup terjangkau namun hanya menyediakan tempat menginap dan sarapan pagi, serta *guest house* baik yang dikelola sebagai usaha swasta (seperti halnya hotel melati) ataupun *mess* yang dikelola oleh perusahaan-perusahaan sebagai tempat menginap bagi para tamu yang ada kaitannya dengan kegiatan atau urusan perusahaan.

Biaya dan kualitas hotel biasanya berdasarkan jangkauan dan tipe pelayanan yang tersedia. Dikarenakan kenaikan besar-besaran dalam turisme di seluruh dunia, selama dekade terakhir abad 20, pendirian hotel terutama yang kecil telah meningkat secara drastis. Untuk pembandingan, sistem penilaian telah diperkenalkan satu sampai lima bintang dan juga tingkat melati di Indonesia yang lebih murah

2.2 Aplikasi Pearlban

Pearlban merupakan singkatan dari "Pencarian Hotel di Bandung". Pearlban merupakan aplikasi *mobile* berbasis android yang digunakan untuk mencari hotel terdekat dengan posisi kita ataupun hotel yang sedang menjadi tujuan kita, dimana aplikasi ini kan menuntun kita ke hotel yang kita tuju karena pada aplikasi ini daitambahkan googlemap.

2.3 Android

Android adalah system operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware*, dan aplikasi kunci. Android

2.3.1. Sejarah dan Perkembangan Android

Pada mulanya terdapat berbagai macam sistem operasi pada perangkat selular, diantaranya sistem operasi Symbian, Microsoft Windsos Mobile, Mobile Linux, iPhone, dan sistem operasi lainnya. Namun diantara sistem operasi yang ada belum mendukung standar dan penerbitan API yang dapat dimanfaatkan secara keseluruhan dan dengan biaya yang murah. Kemudian Google ikut berkecimpung didalamnya dengan platform Android, yang menjanjikan keterbukaan, keterjangkauan, *open source*, dan *framework* berkualitas.

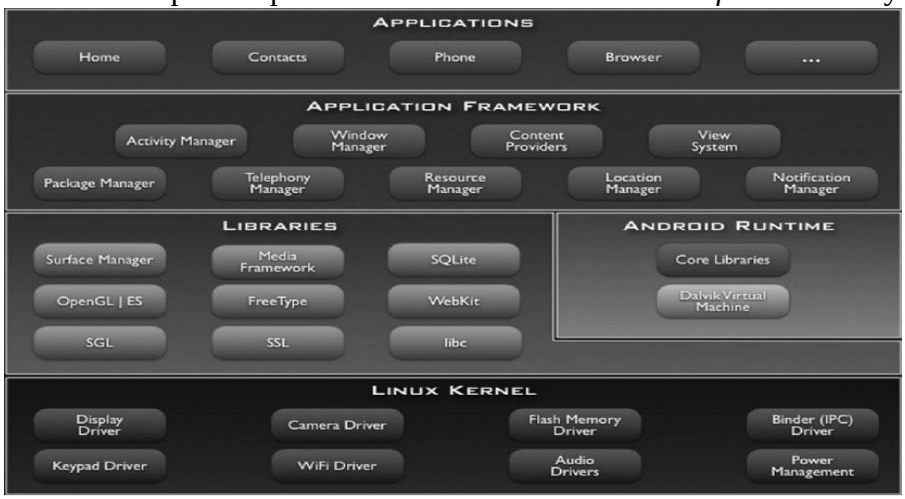
Pada tahun 2005, Google mengakuisisi perusahaan Android Inc. untuk memulai pengembangan platform Android. Dimana terlibat dalam

pengembangan ini Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Pada pertengahan 2007 sekelompok pemimpin industri bersama-sama membentuk aliansi perangkat selular terbuka, Open Handset Alliance (OHA). Bagian dari tujuan aliansi ini adalah berinovasi dengan cepat dan menanggapi kebutuhan konsumen dengan lebih baik, dengan produk awalnya adalah platform Android. Dimana Android dirancang untuk melayani kebutuhan operator telekomunikasi, manufaktur handset, dan pengembang aplikasi. OHA berkomitmen untuk membuat Android *open source* dengan lisensi Apache versi 2.0. [5]

Android pertama kali diluncurkan pada 5 November 2007, dan *smartphone* pertama yang menggunakan sistem operasi Android dikeluarkan oleh T-Mobile dengan sebutan G1 pada bulan September 2008. Hingga saat ini Android telah merilis beberapa versi Android untuk menyempurnakan versi sebelumnya. Selain berdasarkan penomoran, pada setiap versi Android terdapat kode nama berdasarkan nama-nama kue. Hingga saat ini sudah terdapat beberapa versi yang telah diluncurkan, diantaranya: versi 1.5 dirilis pada 30 April 2009 diberi nama *Cupcake*, versi 1.6 dirilis pada 15 September 2009 diberi nama *Donut*, dan versi terakhir 2.0 dirilis pada 26 Oktober 2009 diberi nama *Éclair*.

2.3.2. Anatomi Android

Dalam paket sistem operasi Android tediri dari beberapa unsure. Secara sederhana arsitektur Android merupakan sebuah kernel Linux dan sekumpulan pustaka C/C++ dalam suatu *framework* yang



Gambar 2.1 Anatomi Android

2.3.2.1 Linux Kernel

Android dibangun di atas kernel Linux 2.6. Namun secara keseluruhan Android bukanlah linux, karena dalam Android tidak terdapat paket standar yang dimiliki oleh linux lainnya. Linux merupakan sistem operasi terbuka yang handal dalam manajemen memori dan proses. Oleh karenanya pada Android hanya terdapat beberapa servis yang diperlukan seperti keamanan, manajemen memori, manajemen proses, jaringan dan

driver. Kernel linux menyediakan driver layar, kamera, keypad, WiFi, Flash Memory, audio, dan IPC (*Interprocess Communication*) untuk mengatur aplikasi dan lubang keamanan.

2.3.2.2 Libraries

Android menggunakan beberapa paket pustaka yang terdapat pada C/C++ dengan standar *Berkeley Software Distribution* (BSD) hanya setengah dari yang aslinya untuk tertanam pada kernel Linux. Beberapa pustaka diantaranya:

a. *Media Library* untuk memutar dan merekam berbagai macam format audio dan video.
b. *Surface Manager* untuk mengatur hak akses layer dari berbagai aplikasi.
c. *Graphic Library* termasuk didalamnya *SGL* dan *OpenGL*, untuk tampilan 2D dan 3D.
d. *SQLite* untuk mengatur relasi database yang digunakan pada aplikasi.
e. *SSl* dan *WebKit* untuk browser dan keamanan internet.

Pustaka-pustaka tersebut bukanlah aplikasi yang berjalan sendiri, namun hanya dapat digunakan oleh program yang berada di level atasnya. Sejak versi Android 1.5, pengembang dapat

membuat dan menggunakan pustaka sendiri menggunakan *Native Development Toolkit* (NDK).

2.3.2.3 Android Runtime

Pada Android tertanam paket pustaka inti yang menyediakan sebagian besar fungsi Android. Inilah yang membedakan Android dibandingkan dengan system operasi lain yang juga mengimplementasikan Linux. *Android Runtime* merupakan mesin virtual yang membuat aplikasi Android menjadi lebih tangguh dengan paket pustaka yang telah ada. Dalam Android Runtime terdapat 2 bagian utama, diantaranya:

a. Pustaka Inti, Android dikembangkan melalui bahasa pemrograman Java, tapi Android Runtime bukanlah mesin virtual Java. Pustaka inti Android menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustaka khusus Android.
b. Mesin Virtual Dalvik, Dalvik merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland. Dalvik hanyalah interpreter mesin virtual yang mengeksekusi file dalam format *Dalvik Executable* (*.dex). Dengan format ini Dalvik akan mengoptimalkan efisiensi

penyimpanan dan pengalamatan memori pada file yang dieksekusi. Dalvik berjalan di atas kernel Linux 2.6, dengan fungsi dasar seperti *threading* dan manajemen memori yang terbatas.

2.2.3.4 Aplication Framework

Kerangka aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi Android. Selain itu, juga menyediakan abstraksi generic untuk mengakses perangkat, serta mengatur tampilan *user interface* dan sumber daya aplikasi. Bagian terpenting dalam kerangka aplikasi Android adalah sebagai berikut:

1. *Activity Manager*, berfungsi untuk mengontrol siklus hidup aplikasi dan menjaga keadaan ”*Backstack“* untuk navigasi penggunaan.
2. *Content Providers*, berfungsi untuk merangkum data yang memungkinkan digunakan oleh aplikasi lainnya, seperti daftar nama.
3. *Resuource Manager*, untuk mengatur sumber daya yang ada dalam program. Serta menyediakan akses sumber daya

diluar kode program, seperti karakter, grafik, dan file layout.

4. *Location Manager,* berfungsi untuk memberikan informasi detail mengenai lokasi perangkat Android berada.
5. *Notification Manager*, mencakup berbagai macam peringatan seperti, pesan masuk, janji, dan lain sebagainya yang akan ditampilkan pada *status bar*.

2.3.2.5 Aplication Layer

Puncak dari diagram arsitektur Android adalah lapisan aplikasi dan *widget*. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam *Android runtime* dengan menggunakan kelas dan service yang tersedia pada *framework* aplikasi. Lapisan aplikasi Android sangat berbeda dibandingkan dengan system operasi lainnya. Pada Android semua aplikasi, baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API (*Application Programming Interface*) yang sama.

2.3.3 Komponen Aplikasi

Fitur penting Android adalah bahwa satu aplikasi dapat menggunakan elemen dari aplikasi lain (untuk aplikasi yang memungkinkan). Sebagai contoh, sebuah aplikasi memerlukan fitur *scroller* dan aplikasi lain telah mengembangkan fitur *scroller* yang baik dan memungkinkan aplikasi lain menggunakannya. Maka pengembang tidak perlu lagi mengembangkan hal serupa untuk aplikasinya, cukup menggunakan *scroller* yang telah ada Agar fitur tersebut dapat bekerja, sistem harus dapat menjalankan aplikasi ketika setiap bagian aplikasi itu dibutuhkan, dan pemanggilan objek java untuk bagian itu. Oleh karenanya Android berbeda dari sistem-sistem lain, Android tidak memiliki satu tampilan utama program seperti fungsi main() pada aplikasi lain. Sebaliknya, aplikasi memiliki komponen penting yang memungkinkan sistem untuk memanggil dan menjalankan ketika dibutuhkan.

2.3.3.1 Activities

Activity merupakan bagian yang paling penting dalam sebuah aplikasi, karena *Activity* menyajikan tampilan visual program yang sedang digunakan oleh pengguna. Setiap *Activity* dideklarasikan dalam sebuah kelas yang bertugas untuk menampilkan antarmuka pengguna yang terdiri dari *Views* dan respon terhadap *Event*. Setiap aplikasi memiliki sebuah *activity*

atau lebih. Biasanya pasti akan ada *activity* yang pertama kali tampil ketika aplikasi dijalankan.

Perpindahan antara *activity* dengan *activity* lainnya diatur melalui sistem, dengan memanfaatkan *activity stack.* Keadaan suatu *activity* ditentukan oleh posisinya dalam tumpukan *acitivity*, LIFO (*Last In First Out*) dari semua aplikasi yang sedang berjalan. Bila suatu *activity* baru dimulai, *activity* yang sebelumnya digunakan maka akan dipindahkan ketumpukan paling atas. Jika pengguna ingin menggunakan *activity* sebelumnya, cukup menekan tombol *Back*, atau menutup *activity* yang sedang digunakan, maka *activity* yang berada diatas akan aktif kembali. *Memory Manager* Android menggunakan tumpukkan ini untuk menentukan prioritas aplikasi berdasarkan *activity*, memutuskan untuk mengakhiri suatu aplikasi dan mengambil sumber daya dari aplikasi tersebut.

Ketika *activity* diambil dan disimpan dalam tumpukkan *activity* terdapat 4 kemungkinan kondisi transisi yang akan terjadi :

1) **Active,** setiap *activity* yang berada ditumpukan paling atas, maka dia akan terlihat, terfokus, dan menerima masukkan dari pengguna. Android akan berusaha untuk membuat *activity* aplikasi ini untuk untuk tetap hidup dengan segala

cara, bahkan akan menghentikan *activity* yang berada dibawah tumpukkannya jika diperlukan. Ketika *activity* sedang aktif, maka yang lainnya akan dihentikan sementara.

2) **Paused,** dalam beberapa kasus *activity* akan terlihat tapi tidak terfokus pada kondisi inilah disebut *paused.* Keadaan ini terjadi jika *activity* transparan dan tidak *fullscreen* pada layar. Ketika *activity* dalam keadaan *paused*, dia terlihat *active* namun tidak dapat menerima masukkan dari pengguna. Dalam kasus ekstrim, Android akan menghentikan *activity* dalam keadaan *paused* ini, untuk menunjang sumber daya bagi *activity* yang sedang aktif.

3) Stopped, ketika sebuah *activity* tidak terlihat, maka itulah yang disebut *stopped. Activity* akan tetap berada dalam memori dengan semua keadaan dan informasi yang ada. Namun akan menjadi kandidat utama untuk dieksekusi oleh sistem ketika membutuhkan sumberdaya lebih. Oleh karenanya ketika suatu *activity* dalam kondisi *stopped* maka perlu disimpan data dan kondisi antarmuka saat itu. Karena ketika *activity* telah keluar atau ditutup, maka dia akan menjadi *inactive.*

4) Inactive, kondisi ketika *activity* telah dihentikan dan sebelum dijalankan. *Inactive activity* telah ditiadakan dari

tumpukan *activity* sehingga perlu *restart* ulang agar dapat tampil dan digunakan kembali. Kondisi transisi ini sepenuhnya ditangani oleh manajer memori Android. Android akan memulai menutup aplikasi yang mengandung *activity inactive*, kemudian *stopped activity*, dan dalam kasus luar biasa *paused activity* juga akan di tutup.

2.3.2.2 Services

Suatu *service* tidak memiliki tampilan antarmuka, melainkan berjalan di *background* untuk waktu yang tidak terbatas. Komponen *service* diproses tidak terlihat, memperbarui sumber data dan menampilkan notifikasi. *Service* digunakan untuk melakukan pengolahan data yang perlu terus diproses, bahkan ketika *Activity* tidak aktif atau tidak tampak.

2.3.2.3 Intents

Intens merupakan sebuah mekanisme untuk menggambarkan tindakan tertentu, seperti memilih foto, menampilkan halaman web, dan lain sebagainya. *Intents* tidak selalu dimulai dengan menjalankan aplikasi, namun juga digunakan oleh sistem untuk memberitahukan ke aplikasi bila terjadi suatu hal, misal pesan masuk. *Intents* dapat eksplisit atau implisit, contohnya jika suatu

aplikasi ingin menampilkan *URL*, sistem akan menentukan komponen apa yang dibutuhkan oleh *Intents* tersebut.

2.3.2.4 Broadcast Receivers

Broadcast Receivers merupakan komponen yang sebenarnya tidak melakukan apa-apa kecuali menerima dan bereaksi menyampaikan pemberitahuan. Sebagian besar *Broadcast* berasal dari sistem misalnya, Batre sudah hampir habis, informasi zona waktu telah berubah, atau pengguna telah merubah bahasa *default* pada perangkat. Sama halnya dengan *service, Broadcast Receivers* tidak menampilkan antarmuka pengguna. Namun, *Broadcast Receivers* dapat menggunakan *Notification Manager* untuk memberitahukan sesuatu kepada pengguna.

2.3.2.5 Content Provider

Content Providers digunakan untuk mengelola dan berbagi database. Data dapat disimpan dalam file sistem, dalam database *SQLite*, atau dengan cara lain yang pada prinsipnya sama. Dengan adanya *Content Provider* memungkinkan antar aplikasi untuk saling berbagi data. Komponen ini sangat berguna ketika

sebuah aplikasi membutuhkan data dari aplikasi lain, sehingga mudah dalam penerapannya.

2.3.4 Tipe Aplikasi Android

Terdapat tiga kategori aplikasi pada Android:

1) **Foreground Activity**

   Aplikasi yang hanya dapat dijalankan jika tampil pada layar dan tetap efektif walaupun tidak terlihat. Aplikasi dengan tipe ini pasti mempertimbangkan siklus hidup *activity*, sehingga perpindahan antar *activity* dapat berlangsung dengan lancar.

2) **Background Service**

   Aplikasi yang memiliki interaksi terbatas dengan user, selain dari pengaturan konfigurasi, semua dari prosesnya tidak tidak tampak pada layar. Contohnya aplikasi penyaringan panggilan atau sms auto respon.

3) **Intermittent Activity**

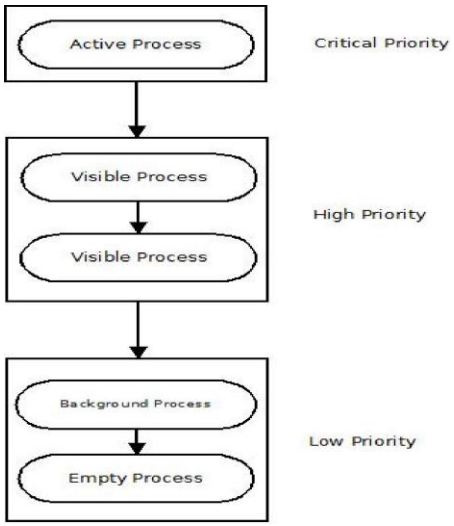
   Aplikasi yang masih membutuhkan beberapa masukkan dari pengguna, namun sebagian sangat efektif jika dijalankan di *background* dan jika diperlukan akan memberi tahu pengguna tentang kondisi tertentu. Contohnya pemutar musik. Untuk aplikasi yang kompleks akan sulit untuk menentukan kategori aplikasi tersebut apalagi aplikasi memiliki ciri-ciri dari semua

kategori. Oleh karenanya perlu pertimbangan bagaimana aplikasi tersebut digunakan dan menentukan kategori aplikasi yang sesuai.

2.3.5 Siklus Hidup Aplikasi Android

Siklus hidup aplikasi Android dikelola oleh sistem, berdasarkan kebutuhan pengguna, sumberdaya yang tersedia, dan sebagainya. Misalnya Pengguna ingin menjalankan browser web, pada akhirnya sistem yang akan menentukan menjalankan aplikasi. Sistem sangat berperan dalam menentukan apakah aplikasi dijalankan, dihentikan sementara, atau dihentikan sama sekali. Jika pengguna ketika itu sedang menjalankan sebuah *Activity*, maka sistem akan memberikan perioritas utama untuk aplikasi yang tersebut. Sebaliknya, jika suatu *Activity* tidak terlihat dan sistem membutuhkan sumber daya yang lebih, maka *Activity* yang prioritas rendah akan ditutup.

Android menjalankan setiap aplikasi dalam proses secara terpisah, yang masing-masing memliki mesin virtual pengolah sendiri, dengan ini melindungi penggunaan memori pada aplikasi. Selain itu juga Android dapat mengontrol aplikasi mana yang layak menjadi prioritas utama. Karenanya Android sangat sensitive dengan siklus hidup aplikasi dan komponen-komponennya. Perlu adanya penanganan terhadap setiap kondisi agar aplikasi menjadi stabil. Gambar dibawah menunjukkan prioritas dari aplikasi.



Gambar 2.2 Prioritas Aplikasi Berdasarkan *Activity*

2.3.6 Kelebihan Android

Sudah banyak *platform* untuk perangkat selular saat ini, termasuk didalamnya Symbian, iPhone, Windows Mobile, BlackBerry, Java Mobile Edition, Linux Mobile (LiM0), dan banyak lagi. Namun ada beberapa hal yang menjadi kelebihan Android. Walaupun beberapa fitur-fitur yang ada telah muncul sebelumnya pada platform lain, Android adalah yang pertama menggabungkan hal seperti berikut :

1. Keterbukaan, Bebas pengembangan tanpa dikenakan biaya terhadap system karena berbasiskan Linux dan *open source*. Pembuat perangkat menyukai hal ini karena dapat membangun *platform* yang sesuai yang diinginkan tanpa harus membayar

royality. Sementara pengembang software menyukai karena Android dapat digunakan diperangkat manapun dan tanpa terikat oleh vendor manapun.

2. Arsitektur komponen dasar Android terinspirasi dari teknologi internet *Mashup*. Bagian dalam sebuah aplikasi dapat digunakan oleh aplikasi lainnya, bahkan dapat diganti dengan komponen lain yang sesuai dengan aplikasi yang dikembangkan.
3. Banyak dukungan service, kemudahan dalam menggunakan berbagai macam layanan pada aplikasi seperti penggunaan layanan pencarian lokasi, database SQL, browser dan penggunaan peta. Semua itu sudah tertanam pada Android sehingga memudahkan dalam pengembangan aplikasi.
4. Siklus hidup aplikasi diatur secara otomatis, setiap program terjaga antara satu sama lain oleh berbagai lapisan keamanan, sehingga kerja system menjadi lebih stabil. Pengguna tak perlu kawatir dalam menggunakan aplikasi pada perangkat yang memorinya terbatas.
5. Dukungan grafis dan suarat terbaik, dengan adanya dukungan 2D grafis dan animasi yang diilhami oleh *Flash* menyatu dalam 3D menggunakan *OpenGL* memungkinkan membuat aplikasi maupun game yang berbeda.

6. Portabilitas aplikasi, aplikasi dapat digunakan pada perangkat yang ada saat ini maupun yang akan datang. Semua program ditulis dengan menggunakan bahas pemrograman Java dan dieksekusi oleh mesin virtual Dalvik, sehingga kode program portabel antara ARM, X86, dan arsitektur lainnya. Sama halnya dengan dukungan masukan seperti penggunaan *Keyboard,* layar sentuh, *trackball* dan resolusi layar semua dapat disesuaikan dengan program.

2.4 Objek Oriented

Obyek dalam *'software analysis & design'* adalah sesuatu berupa konsep (*concept*), benda (*thing*), dan sesuatu yang membedakannya dengan lingkungannya. Secara sederhana obyek adalah mobil, manusia, *alarm* dan lainlainnya. Tapi obyek dapat pula merupakan sesuatu yang abstrak yang hidup didalam sistem seperti tabel, *database*, *event*, *system messages.*

Obyek dikenali dari keadaannya dan juga operasinya. Sebagai contoh sebuah mobil dikenali dari warnanya, bentuknya, sedangkan manusia dari suaranya. Ciriciri ini yang akan membedakan obyek tersebut dari obyek lainnya.

Alasan mengapa saat ini pendekatan dalam pengembangan software dengan *object-oriented,* pertama adalah *scalability* dimana obyek lebih mudah

dipakai untuk menggambarkan sistem yang besar dan komplek. Kedua *dynamic modeling*, adalah dapat dipakai untuk permodelan sistem dinamis dan *real time*.

2.4.1 UML

Dalam suatu proses pengembangan software, analisa dan rancangan telah merupakan terminologi yang sangat tua. Pada saat masalah ditelusuri dan spesifikasi dinegoisasikan, dapat dikatakan kita berada pada tahap rancangan. Merancang adalah menemukan suatu cara untuk menyelesaikan masalah, salah satu tool / model untuk merancang pengembangan software yang berbasis object oriented adalah UML.

UML adalah kependekan dari Unified Modeling Language yang merupakan suatu cara untuk menyelesaikan suatu masalah dengan mendeskripsikannya yang telah menjadi standar dalam dunia industri untuk memvisualisasikan, merancang dan mendokumentasikan sistem perangkat lunak. Dengan menggunakan UML kita dapat membangun model untuk segala bentuk dan jenis aplikasi perangkat lunak, yang mana aplikasi yang dibangun dapat berjalan pada perangkat lunak dengan sistem operasi dan jaringan apapun. Penggunaan UML dalam membangun model suatu sistem yang berbentuk diagram-diagram memudahkan baik pengguna maupun programmer dalam memahami ururtan dalam sebuah sistem yang dirancang.

Dalam membangun suatu model perangkat lunak dengan UML, digunakan bentuk-bentuk diagram atau symbol untuk merepresentasikan elemen-elemen dalam sistem. Bentuk diagram yang digunakan untuk merepresentasikannya adalah sebagai berikut :

1. Use-case Diagram

   Use case diagram menggambarkan fungsionalitas yang diharapkan darisebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, danbukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksiantara aktor dengan sistem.

2. Class Diagram

   Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkansebuah objek dan merupakan inti dari pengembangan dan desain berorientasiobjek. Class menggambarkan keadaan (atribut/properti) suatu sistem,sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut(metoda/fungsi).

3. State Diagram

   Statechart diagram menggambarkan transisi dan perubahan keadaan (darisatu state ke state lainnya) suatu objek pada sistem sebagai akibat dari stimuliyang diterima. Pada umumnya statechart diagram menggambarkan classtertentu (satu class dapat memiliki lebih dari satu statechart diagram)

4. Sequence diagram

   Sequence diagram menggambarkan interaksi antar objek di dalam dan disekitar sistem (termasuk pengguna, display, dan sebagainya) berupa messageyang digambarkan terhadap waktu. Sequence diagram terdiri atar dimensivertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

5. Collaboration Diagram

   Collaboration diagram juga menggambarkan interaksi antar objek sepertisequence diagram, tetapi lebih menekankan pada peran masing-masing objekdan bukan pada waktu penyampaian message. Setiap message memilikisequence number, di mana message dari level tertinggi memiliki nomor 1.Messages dari level yang sama memiliki prefiks yang sama.

6. Activity Diagram

   Activity diagram menggambarkan berbagai alir aktivitas dalam sistemyang sedang dirancang, bagaimana masing-masing alir berawal, decision yangmungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapatmenggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.Activity diagram merupakan state diagram khusus, di mana sebagian besar

   state adalah action dan sebagian besar transisi di-trigger oleh selesainya statesebelumnya (internal processing). Oleh karena itu

activity diagram tidakmenggambarkan behaviour internal sebuah sistem (dan interaksi antarsubsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur jalur aktivitas dari level atas secara umum.

7. Component Diagram

   Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (dependency) di antaranya.Komponen piranti lunak adalah modul berisi code, baik berisi source codemaupun binary code, baik library maupun executable, baik yang muncul padacompile time, link time, maupun run time. Umumnya komponen terbentuk dari beberapa class dan/atau package, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa interface, yaitu kumpulanlayanan yang disediakan sebuah komponen untuk komponen lain.

8. Deployment Diagram

   Deployment/physical diagram menggambarkan detail bagaimana komponen di-deploy dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.Sebuah node adalah

server, workstation, atau piranti keras lain yang digunakan untuk men-deploy komponen dalam lingkungan sebenarnya.Hubungan antar node (misalnya TCP/IP) dan requirement dapat juga di definisikan dalam diagram ini.

2.5 PHP

PHP adalah singkatan dari "PHP: Hypertext Preprocessor", yang merupakan sebuah bahasa scripting yang terpasang pada HTML. Sebagian besar sintaks mirip dengan bahasa C, Java dan Perl, ditambah beberapa fungsi PHP yang spesifik.

Tujuan utama penggunaan bahasa ini adalah untuk memungkinkan perancang web menulis halaman web dinamik dengan cepat.

2.5.1 Hubungan PHP dengan HTML

Halaman web biasanya disusun dari kode-kode html yang disimpan dalam sebuah file berekstensi .html. File html ini dikirimkan oleh server (atau file) ke browser,

Kemudian browser menerjemahkan kode-kode tersebut sehingga menghasilkan suatu tampilan yang indah. Lain halnya dengan program php, program ini harus diterjemahkan oleh web-server sehingga menghasilkan kode html yang dikirim ke browser agar dapat ditampilkan. Program ini dapat berdiri sendiri ataupun disisipkan di antara kode-kode

html sehingga dapat langsung ditampilkan bersama dengan kode-kode html tersebut.

Program php dapat ditambahkan dengan mengapit program tersebut di antara tanda . Tanda-tanda tersebut biasanya disebut tanda untuk escaping (kabur) dari kode html. File html yang telah dibubuhi program php harus diganti ekstensi-nya menjadi .php3 atau php.

PHP merupakan bahasa pemograman web yang bersifat server-side HTML=embedded scripting, di mana script-nya menyatu dengan HTML dan berada si server. Artinya adalah sintaks dan perintah-perintah yang kita berikan akan sepenuhnya dijalankan di server tetapi disertakan HTML biasa. PHP dikenal sebgai bahasa scripting yang menyatu dengan tag HTML, dieksekusi di server dan digunakan untuk membuat halaman web yang dinamis seperti ASP (Active Server Pages) dan JSP (Java Server Pages).

2.5.2 Sejarah PHP

PHP pertama kali dibuat oleh Rasmus Lerdroft, seorang programmer. Semula PHP digunakannya untuk menghitung jumlah pengunjung di dalam webnya. Kemudian ia mengeluarkan Personal Home Page Tools versi 1.0 secara gratis. Versi ini pertama kali keluar pada tahun 1995. Isinya adalah sekumpulan script PERL yang dibuatnya untuk membuat halaman webnya menjadi dinamis. Kemudian pada tahun 1996 ia mengeluarkan PHP versi

2.0 yang kemampuannya telah dapat mengakses database dan dapat terintegrasi dengan HTML.

Pada tahun 1998 tepatnya pada tanggal 6 Juni 1998 keluarlah PHP versi 3.0 yang dikeluarkan oleh Rasmus sendiri bersama kelompok pengembang softwarenya.Versi terbaru, yaitu PHP 4.0 keluar pada tanggal 22 Mei 2000 merupakan versi yang lebih lengkap lagi dibandingkan dengan versi sebelumnya.

Perubahan yang paling mendasar pada PHP 4.0 adalah terintegrasinya Zend Engine yang dibuat oleh Zend Suraski dan Andi Gutmans yang merupakan penyempurnaan dari PHP scripting engine. Yang lainnya adalah build in HTTP session, tidak lagi menggunakan library tambahan seperti pada PHP. Tujuan dari bahasa scripting ini adalah untuk membuat aplikasi-aplikasi yang dijalankan di atas teknologi web.

Dalam hal ini, aplikasi pada umumnya akan memberikan hasil pada web browser, tetapi prosesnya secara keseluruhan dijalankan web server. Seluruh aplikasi berbasis web dapat dibuat dengan PHP. Namun kekuatan yang paling utama PHP adalah pada konektivitasnya dengan system database di dalam web. Sistem database yang dapat didukung oleh PHP adalah Oracle, MySQL, Sybase, PostgreSQL, dan lainnya.

PHP dapat berjalan di berbagai system operasi seperti windows 98/NT, UNIX/LINUX, solaris maupun macintosh PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP bernama FI (Form

Interpreted). Pada saat tersebut PHP adalah sekumpulan script yang digunakan untuk mengolah data form dari web.

Perkembangan selanjutnya adalah Rasmus melepaskan kode sumber tersebut dan menamakannya PHP/FI, pada saat tersebut kepanjangan dari PHP/FI adalah Personal Home Page/Form Interpreter. Dengan pelepasan kode sumber ini menjadi open source, maka banyak programmer yang tertarik untuk ikut mengembangkan PHP.

Pada November 1997, dirilis PHP/FI 2.0. Pada rilis ini interpreter sudah diimplementasikan dalam C. Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan PHP/FI secara signifikan.

Pada tahun 1997, sebuah perusahaan bernama Zend, menulis ulang interpreter PHP menjadi lebih bersih, lebih baik dan lebih cepat. Kemudian pada Juni 1998 perusahaan tersebut merilis interpreter baru untuk PHP dan meresmikan nama rilis tersebut menjadi PHP 3.0.

Pada pertengahan tahun 1999, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang paling banyak dipakai. Versi ini banyak dipakai sebab versi ini mampu dipakai untuk membangun aplikasi web kompleks tetapi tetap memiliki kecepatan proses dan stabilitas yang tinggi.

Pada Juni 2004, Zend merilis PHP 5.0. Versi ini adalah versi mutakhir dari PHP. Dalam versi ini, inti dari interpreter PHP mengalami perubahan besar. Dalam versi ini juga dikenalkan model pemrograman berorientasi

objek baru untuk menjawab perkembangan bahas pemrograman kearah pemrograman berorientasi objek.

2.5.3 Kelebihan PHP dari Bahasa Pemrograman Lain

Berikut adalah kelebihan - kelebihan PHP dari bahasa pemrograman lain :

a Bahasa pemrograman php adalah sebuah bahasa script yang tidak melakukan sebuah kompilasi dalam penggunaanya.

b Web Server yang mendukung php dapat ditemukan dimana - mana dari mulai IIS sampai dengan apache, dengan configurasi yang relatif mudah.

c Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan developer yang siap membantu dalam pengembangan.

d Dalam sisi pemahamanan, php adalah bahasa scripting yang paling mudah karena referensi yang banyak.

e PHP adalah bahasa open source yang dapat digunakan di berbagai mesin (linux, unix, windows) dan dapat dijalankan secara runtime melalui console serta juga dapat menjalankan perintah-perintah system